Hunting Sybils in Participatory Mobile Consensus-Based Networks

Nickolai Verchok
nverchok@eng.ucsd.edu
Computer Science and Engineering
University of California, San Diego

Alex Orailoğlu
alex@eng.ucsd.edu
Computer Science and Engineering
University of California, San Diego

ABSTRACT

We focus on detecting adversarial non-existent nodes, Sybils, in anonymized participatory mobile networks where nodes support both a node-to-server and peer-to-peer connection capabilities. As data-driven decisions within such networks typically rely on local consensuses, they are susceptible to adversarial injection attacks which impersonate honest nodes and overpower local data through forgery.

First, we propose a scheme wherein nodes validate each other's presence through local peer-to-peer communication. We then observe a fundamental information asymmetry between Sybils and honest nodes, and argue that conventional Sybil detection techniques fail to exploit it. Thereupon, we propose a novel Sybil detection technique tailored to utilizing claimed location data, introduce a probabilistic framework for the problem, and design the statistical approach for finding Sybils. Finally, we compare our detection algorithm with existing methods through complex simulated Sybil scenarios.

CCS CONCEPTS

• **Security and privacy → Intrusion detection systems**; • **Hardware → Sensor devices and platforms**;

KEYWORDS

Sybil detection; location validation; participatory networks; security; probabilistic model

ACM Reference format:
Nickolai Verchok and Alex Orailoğlu. 2020. Hunting Sybils in Participatory Mobile Consensus-Based Networks. In *Proceedings of 15th ACM Asia Conference on Computer and Communications Security, Taipei, Taiwan, October 5–9, 2020 (ASIA CCS '20),* 12 pages.
https://doi.org/10.1145/3320269.3372200

1 INTRODUCTION

Participatory crowdsourced sensing is becoming an increasingly attractive paradigm[1, 2] for information gathering in critical applications; inexpensive sensors may be distributed to large crowds in order to provide wide, dynamic coverage, relaying their data to one centralized Server. Recent advances in smartphone technology have immensely benefitted the approach; sensors may nowadays be smartphone-mountable[3, 4] and offload their computation and communication to the device. This has been used to bundle sensor data with smartphone-provided timestamps and location information, facilitating the Server's decision-making.

In many contexts, users may participate in the network *anonymously* in order to preserve the privacy of their time-stamped location data[5, 6]. Specifically, messages sent from a participating user (one who voluntarily attaches the sensor to their smartphone) would not contain information that may personally identify them, but would contain an unforgeable identifier for the sensor (to filter out bogus messages). This anonymity is crucial for ensuring that sufficient numbers of participants willingly join and become *nodes* in the network.

Unfortunately, this anonymity allows an adversary to participate in the network and employ deliberate disinformation, corrupting the Server's data and misguiding its decision-making. While this problem is fundamental to anonymous systems, it is greatly exacerbated in this case by the fact that messages do not even have to originate from physical devices[7]. An adversary in possession of multiple sensors may forge fake messages and inject them into the network at a purely-software level. In this manner, a far-away adversary may create a multitude of fake immaterial nodes, referred to as Sybils[8], that pretend to be sending particular sensor readings from particular locations in order to mislead the Server.

A further complication arises from an economical sacrifice of sensor quality for sensor quantity, where systems may prefer a higher number of cheaper, less accurate sensors. While potentially creating more data sources, the data from each source becomes less reliable as false positives and false negatives compound. A typical countermeasure is for the Server to act upon *local consensuses of nearby nodes*[9] instead of individual node data, thereby introducing some robustness into the decision-making process.

Such consensus evaluations constitute the prime targets for Sybil attacks however, as the Server may be tricked into dismissing honest node data as "sensor noise" in favor of a locally-overwhelming mass of conflicting Sybil data. While significant countermeasures have been developed for detecting Sybils within the software realm, techniques for detecting Sybils within Cyber-Physical Systems (CPS) are still in their infancy, with several unresolved technical challenges rendering them inadequate for practical implementation. We believe that until viable solutions to the Sybil problem are identified, the immense power and cost-efficiency of participatory crowdsourced systems will remain untapped.

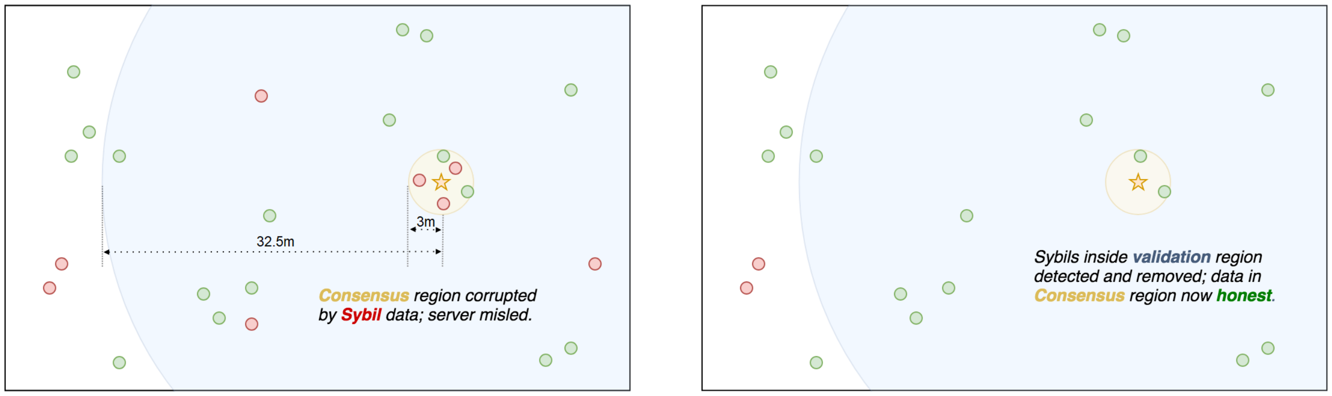
32.5m
3m
Consensus region corrupted by Sybil data; server misled.
Sybils inside validation region detected and removed; data in Consensus region now honest.

Figure 1: An example application. Sybil nodes (red) mislead the Server by sending forged data allegedly from within the small *data consensus* region (yellow; radius determined by e.g. sensor quality). Sufficient honest nodes (green) within the much-larger enclosing *location validation* region (blue; radius determined by P2P communication medium) are used to collectively detect and eliminate Sybils, thereby purifying consensus data.

Our contribution is twofold. We first devise a novel location validation procedure that addresses the hitherto unresolved technical challenges, and constructs a trustworthy and informative *proximity graph* which may be analyzed in order to detect Sybils. The graph is formed by instructing nodes to communicate to each other via a range-constrained peer-to-peer (P2P) medium, with formed pairwise connections acting as subjective, potentially-contradictory evidence of physical presence for the nodes. Secondly, we argue that standard graph-based Sybil detection techniques are not designed to make use of the additional location information inherently embedded in the proximity graph, and present our own statistical framework for analyzing the graph and detecting Sybils. An example application of our technique is shown in Fig. 1.

2 PAPER OUTLINE

The paper is organized as follows. In Section 3 we outline the Sybil problem and contextualize it within Cyber-Physical Systems, with a particular focus on participatory mobile sensor networks. We present the current countermeasure techniques, identify their limitations, and outline technical challenges we set out to solve.

In Section 4 we propose our graph construction scheme. We describe the peer-to-peer communication procedure, detail its internal algorithm for generating nodewise action sequences within the procedure, and outline the scheme's advantages.

In Section 5 we analyze the proximity graph. First, we present a probabilistic framework for the interpretation of the proximity graph. Next, we describe how the trustworthiness of a node may be assessed. Finally, we describe the statistical techniques necessary for analysis, and describe two algorithms for detecting Sybils tailored for our proximity graph.

In Section 6 we present our simulation framework and discuss our results. We demonstrate adversarial strategies that confuse standard graph-based detection algorithms by removing any obvious Sybil community structure, and show the resilience of our own approach to such evasive maneuvers.

We conclude the paper in Section 7, discussing the robustness, flexibility, and extensibility of our approach.

3 SYBIL DETECTION OVERVIEW

The Sybil problem has been widespread in the domain of social networks, such as Facebook and YouTube, where Sybils (fake accounts) transmit spam, malware, or influence popularity rankings and voting systems. Due to their inherent emphasis on pairwise relationships between the nodes, social networks lend themselves nicely to **graph representations**, with the edges representing those relationships (e.g. undirected Facebook friendship, directed Twitter following, etc).

As such, many **Graph-based Sybil Detection (GSD)** algorithms have been developed as countermeasures, such as SybilGuard[10], SybilRank[11], and many more. A detailed mathematical analysis and comparison of the most widely-used algorithms may be found in this paper[12]. At an intuitive level, these algorithms typically find *communities* within graphs, working on the premise of the preponderance of honest–honest and Sybil–Sybil edges over edges between honest nodes and Sybils.

3.1 Sybils in Cyber-Physical Systems

In participatory cyber-physical systems, however, there are fundamental differences that add to the complexity of the Sybil detection problem. As these systems are typically concerned with extracting data from individual participating nodes rather than defining relationships *between* them, **there typically is no natural graph to perform such Sybil detection on**, precluding the use of GSD algorithms (unlike in the Social Network domain).

Consider two recent participatory mobile sensing systems developing within the military sector [13, 14]. Their main idea is to distribute cheap and disposable sensors to ordinary pedestrians in order to periodically scan their vicinity for traces of explosive chemicals. The sensors would attach to smartphones which would transmit their readings to a centralized Server, supplementing them with a timestamp and location data.

This setup places the Server at the mercy of the smartphones' honesty, as location-spoofed messages would be indistinguishable from honest ones to the Server. Without Sybil detection, an adversary would easily be able to emulate an entire cluster of seemingly-real Sybils and use it to mislead the Server. This would be detrimental, as such an undiscovered Sybil cluster could conceal the presence of a bomb by sending an overwhelming amount of benign sensor readings from its location, or waste police resources by claiming false alarms.

The physical, real-world nature of these systems also introduces a sharp distinction between the adversarial nodes that correspond to physical devices attempting to mislead the Server, and adversarial nodes *emulated purely through software* without a physical counterpart. This creates the trichotomy of node types which we will henceforth refer to in the following way:

- **Honest node**: honest participant; physical device
- **Malicious node**: adversarial participant; physical device
- **Sybil node**: adversarial participant; emulated (no device)

This work focuses on **detecting the Sybils specifically**, as Sybils allow an adversary to leverage their power at almost no cost (as software simulation is cheaper than having physical agents on the field), and at low risk (as the adversary will not be subject to physical security on-site).

3.2 Current State of Countermeasures

There are two primary approaches to imbuing such systems with Sybil detection capabilities. The first approach is to physically deploy a trusted communication infrastructure, which will additionally verify node proximity or triangulate their location. These proposals predominantly come from the domain of VANETs, and make use of distance-based authentication schemes [16, 17], or triangulation[18] with directional antennae. While these proposals may be feasible given a complete overhaul of our city streets with the deployment of millions of trusted road-side units or antennae, their practicality in the near term remains unclear.

The second approach is to crowdsource the validation process itself, requiring no additional infrastructure beyond the regular smartphones already available to the participating nodes. The key idea is to let nodes vouch for each other's presence, form a graph of evidence of which nodes can attest to the physical presence of which other nodes, and perform Sybil detection on this graph. Techniques for doing this have been proposed in [7, 19] which we detail next, as they are most similar to our proposed approach.

3.3 Peer-to-peer Proximity Graphs

Range-restricted P2P communication may be exploited in order to have nodes *validate their claimed physical presence to one another*. In previously proposed schemes, any pair of nodes whose claimed locations are within a threshold ($80m$) are instructed to establish a pairwise connection and perform a key exchange. This is done by one node broadcasting its server-provided *one-time secret key* over the P2P medium (e.g. WiFi Direct or Bluetooth) to the other, which in turn reports it to the Server, or notifies it that the connection had failed. Upon a success, a new undirected edge between these two nodes is permanently added to the proximity graph; upon a failure, nothing is done.

Assuming perfect P2P communication and an unsophisticated adversary, all honest–honest and honest–malicious edges would form, as the honest nodes would indeed be within the communication range. Sybil–Sybil and Sybil–malicious edges could freely form as well (at the discretion of the Adversary), as the key exchanges could happen through external communication, thereby having pair of Sybils *pretending* to have seen each other. Sybil–honest edges would never form however, as Sybils do not correspond to physical devices and therefore have no way to receive or transmit signals on the P2P medium.

As these edges individually form one-by-one, the tight community characteristics of the Sybils would reveal themselves over time. The Sybils could subsequently be identified by GSD algorithms, so long as there is a sufficiently-larger community of honest nodes.

3.4 Technical Challenges

We outline a number of shortcomings of the current pairwise proximity scheme that render it impractical, ineffective, and vulnerable to circumvention:

(1) **Short Longevity: It severely strains devices.** The *one-by-one* formation of pairwise connections results in a constant usage of every device's WiFi transceiver, perpetually disrupting normal user service while additionally draining power. This inconvenience may frustrate new users from participating in the system and reduce the longevity of the devices.

(2) **Low Privacy: It requires accumulated historical data.** Even anonymous records may succumb to statistical analysis. As a specific example, many short paths between two nodes in the proximity graphs (as constructed in [7]) may indicate that those two nodes (and hence users) remained in close proximity for a prolonged period of time.

(3) **Weak Security: It is susceptible to Sybil infiltration.** We believe that malicious nodes have incredibly potent capabilities in helping Sybils stay undetected that have hitherto been unexamined. We outline 3 particular techniques an adversary may employ to obscure their Sybil nodes:

    Impersonation: Anytime a Sybil is instructed to broadcast to an honest node, a nearby malicious node may *impersonate* the Sybil and *broadcast on the Sybil's behalf*, unbeknownst to the honest node and the Server.

    Circulation: Anytime a Sybil is instructed to receive data from an honest node, a nearby malicious node may listen on the Sybil's behalf and *circulate* the data to the Sybil, allowing it to claim *to have received the data itself*.

    Dissociation: Sybils and malicious nodes may choose to *dissociate* from each other to *thwart community detection algorithms*, artificially reducing their own interconnectivity by failing the communication process.

    These techniques allow an adversary to gradually establish Sybil–honest edges while also reducing the number of Sybil–Sybil and Sybil–malicious edges, to the final effect of reducing the Sybil clustering behavior. With the clustering reduced, the detection capabilities of GSD algorithms are thwarted as there is no longer a clear "Sybil community". Such an infiltration may lead to detrimental consequences: Sybil nodes with sufficient accumulated trust would be able to not only mislead the data aspect of the network, but to outright bully-out freshly-joined honest nodes who have not yet had sufficient time to establish their credibility.

3.5 Our Approach

We address these technical challenges in two steps. First, we develop a novel P2P communication scheme to construct a proximity graph, described in Section 4. Our scheme generates an entire fully-connected directed graph in one shot in **logarithmic time** (in the number of nodes) to greatly reduce device strain and remove necessity of accumulating historical data.

In Section 5 we design a novel detection scheme around a probabilistic model for edge successes and failures. We argue that the trustworthiness of a node is captured by the observed log-likelihood value of the collection of incoming edges into that node. As this observed value is a realization of a random variable, we set out to compute the variable's distribution in order to evaluate the probability of observing such a log-likelihood value (its left-tail *p-value*). To overcome the exponential direct-computation of the distribution, we show that it is well-approximated **in linear time** by the Gaussian through Lyapunov's version of the Central Limit Theorem which allows for differently-distributed summands. We verify that the convergence is excellent even at small numbers of nodes via explicit computation. Finally, we present two detection algorithms based on these nodewise *p-values* which harness the location information embedded in the proximity graph, leading to significantly superior detection performance, demonstrated in Section 6.

4 PROXIMITY GRAPH CONSTRUCTION

To construct an informative proximity graph that could then be analyzed to identify Sybils, we propose the following scheme, applicable to a general class of P2P-capable CPSs. As a meta-parameter, the P2P medium is chosen appropriately for the target application, such as WiFi Direct for smartphone-based networks. With modern smartphones in mind, we make the following device assumptions:

(1) Devices have homogeneous communication capabilities
(2) Devices can **either** broadcast (trasmit) **or** scan (receive)
(3) Devices get frequent location updates (every few seconds)

4.1 Communication Procedure Overview

The procedure involves a series of $2\lceil log_2(N)\rceil$ discrete, synchronized, α-second-long rounds of dense broadcast-communication between all of the N nodes that are initially within a radius $\mathcal{R}$ around a specified center location, with each round lasting for α seconds[1]. Nodes take turns to declare their own presence and tabulate the presence of others in broadcast fashion, and subsequently communicate these results to the Server, which proceeds to form a proximity graph of *who-saw-who*. The scheme consists of the following steps:

[1]The radius $\mathcal{R}$ and time-per-round α are chosen to yield sufficient participants while ensuring reasonable connectivity between every pair in the pairs' respective rounds. For WiFi Direct, our default choices are $\mathcal{R} = 32.5$m and $\alpha = 2$sec.

(1) The Server chooses to initiate the procedure at a particular time centered at a particular location
(2) All nodes within $\mathcal{R}$ radius of a center become participants
(3) Server generates sequences of roundwise listen/broadcast states and random broadcasting keys for every node
(4) Server distributes state sequences and keys to each node; in each round, each node either listens or broadcasts its key
(5) Synchronized, nodes initiate communication, listening or broadcasting their keys via the P2P communication medium
(6) Nodes accumulate lists of keys they *actually* see for rounds wherein they listen
(7) Nodes report their *seen-lists* to the Server
(8) Server maps keys in the *seen-lists* to their respective nodes
(9) Server constructs a proximity graph; entries [present in | missing from] the mapped *seen-lists* produce [successful | failed] edges *from* the listening *to* the broadcasting nodes

4.2 Nodewise State Sequence Generation

The roundwise state assignments for each node (i.e. whether it should listen, remain idle, or broadcast a specific key) are obtained using the GEN_STATE_SEQUENCES algorithm outlined below.

This algorithm returns a 2D *num_nodes* × *num_rounds* string array, where each row encodes the sequence of state assignments for a given node, and each entry of this sequence is either *"idle"*, *"listen"*, or some unique key (implying broadcasting). This array is initially filled with *"idle"* and is then recursively filled out. The runtime is trivially fast and immaterial to the overall approach.

The unique, practically-unguessable keys for the broadcasting states are generated on the fly with any suitable GEN_KEY() function.

Algorithm 1 Nodewise State Sequence Generation

```
procedure GEN_STATE_SEQUENCES(N)
    $num\_rounds \leftarrow 2 * \lceil log_2(N) \rceil$
    $nsseqs \leftarrow N \times num\_rounds$ string array
    FILL($nsseq$, "idle")
    GEN_ROUNDWISE_STATES($nsseqs, 0, 0, N$)
    return $nsseqs$

procedure GEN_ROUNDWISE_STATES($nsseqs, depth, bgn, end$)
    if $end - bgn > 1$ then
        $mid \leftarrow \lfloor (bgn + end)/2 \rfloor$
        for $i = bgn, i < mid$ do
            $nsseqs[i][depth]$ = GEN_KEY()
            $nsseqs[i][depth + 1]$ = "listen"
        for $i = mid, i < end$ do
            $nsseqs[i][depth]$ = "listen"
            $nsseqs[i][depth + 1]$ = GEN_KEY()
        if $depth+2 < num\_rounds$ then
            GEN_NODEWISE_STATES($nsseq, depth+2, bgn, mid$)
            GEN_NODEWISE_STATES($nsseq, depth+2, mid, end$)
```

4.3 Execution of Communication

Every two rounds, the state assignments are constructed such as to split the nodes into two approximately-equal groups of listeners and broadcasters. In the first round of the two, the first half of the nodes listen while the second half broadcast. In the second round of the two, the nodes swap roles. Thus after every pair of rounds completes, every node will have had a chance to both *see* and *be seen* by every other node from the other group. As will be demonstrated in Section 4.4, these node-to-node connections will be represented by a proximity graph.

After the very first pair of rounds, there are effectively 2 groups of nodes which have full bipartite connectivity, but are lacking in internal connections (as the nodes within each group were all listening or all broadcasting in both rounds).

In the following pair of rounds, these 2 groups are further split into 2 subgroups each, and the procedure recurses to conclusion, until every node has had a chance to *"listen"* during every other node's "broadcasting" state. At this point, assuming no communication failures, the nodes have full N-partite and complete bidirectional connectivity, with every directed edge denoting whether the source node *has seen* the broadcast of the destination node and can thereby *vouch for the latter's existence.*

In this scheme, **all communication is done in open broadcast form**, implying that *no node-to-node channels actually have to be established.* In the example of using WiFi Direct, broadcasting nodes would transmit tailored 802.11 beacon frame packets with SSID fields housing the secret keys. This approach **avoids** establishing N^2 pairwise communication channels (each with further unnecessary handshake procedures), allowing instead to complete the entire procedure in $O(log(N))$ time.

Additionally, nodes receive regular smartphone location updates every α seconds (once every round) and incorporate this roundwise location data into their *seen-lists*, allowing the Server to appropriately model the likelihoods of random connection failures according to a distance model, further discussed in Section 5.

4.4 Forming the Proximity Graph

After assigning every participating node with its respective sequence of states derived using Algorithm 1 and initiating the communication procedure, the Server gets back lists of *seen-keys*, one for every node. The Server then compares them against the hypothetical connections that *could have occurred* given the listener-broadcaster pairs in every round, and will construct a proximity graph G as follows.

The graph $G = (V, E)$ is a collection of N nodes: $V = \{v_1, ..., v_N\}$, and a complete set of edges: $E = V \times V$.

Node **locations** are given, for relevant rounds, by the function $loc : V \times \mathbb{Z}^+ \longrightarrow \mathbb{R}^2$, where the x, y coordinates claimed by node $v \in V$ in round number $r \in \mathbb{Z}^+$ are given by $loc(v, r) = (x, y)$.

A directed edge from v_i to v_j is given by $e_{i,j}$, and $res : E \longrightarrow \{S, F\}$ denotes whether the **result** was a success (S), meaning that v_i *saw* v_j (v_j's key is in v_i's *seen-list*), or a failure (F).

The function $rnd : E \longrightarrow \mathbb{Z}^+$ encodes the **round** wherein the edge was established. The length of an edge $e_{i,j}$ (i.e. Euclidean distance between v_i and v_j when the edge formed) is given by $dist : E \longrightarrow \mathbb{R}^+$, where: $dist(e_{i,j}) = \left\| loc(v_i, rnd(e_{i,j})) - loc(v_j, rnd(e_{i,j})) \right\|_2$.

4.5 An Example

Suppose that the Server decides to initiate the Location Validation procedure at a particular point and time, finding nodes $\{A, B, C, D\}$ to be within the predefined radius. First, the server will distribute keys $\{k1, k5\}$ to A, $\{k4, k7\}$ to B, $\{k3, k6\}$ to C, and $\{k2, k8\}$ to D. Then, it will synchronize the nodes, which will initiate the communication.

In round 1, A and D will broadcast while B and C will listen and record the keys they see. In round 2, the roles will reverse. In round 3, A and C will broadcast, and in round 4, the roles will again reverse. This is demonstrated by Fig. 2 below. After the 4 rounds, each node will have had a chance to see a key belonging to each other node, and they will all report their *seen-lists* to the Server.

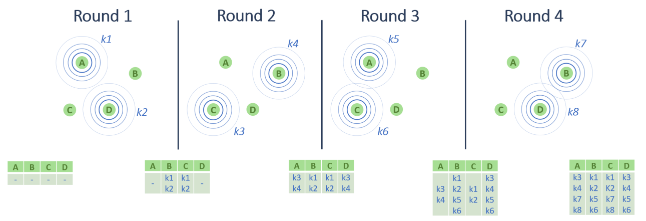

Figure 2: Communication procedure for 4 nodes: A, B, C, D. Concentric circles represent broadcasting. The *seen-lists* are shown inbetween the rounds and are showing all *potential* connections. At the end, every node has had a chance to listen to (at least) one broadcast from every other node.

Suppose that A *did not* see $k8$, and that C is a Sybil node that could neither have seen any keys, nor could have any of its own keys seen by others. This situation would have the Server produce the following proximity graph shown in Fig. 3 below:

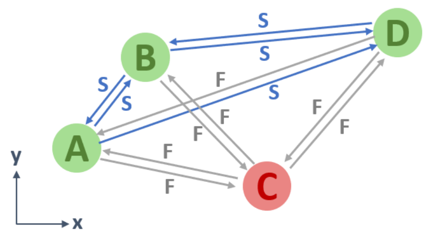

Figure 3: Proximity graph generated from the communication procedure results, assuming C is a Sybil (causing its edge failures), and the $D \rightarrow A$ edge failed by pure chance. Nodes are assumed to be stationary for simplicity.

4.6 Advantages

This scheme is designed to address the shortcomings discussed prior in the following ways:

(1) **Devices are involved *sparingly*.** Perpetual attempts at establishing connections hog the WiFi transceiver, drain power, and are likely to irritate the user. The quick, sub-1-minute, one-shot graph formation on the other hand improves the overall system longevity and tolerability.

(2) **Historical data is *not* necessary.** New nodes entering the overarching crowdsourced system do not have to accumulate trust, since they may immediately be full participants of any proximity-graph-based Sybil detection procedure. The lack of historical record-keeping additionally carries a privacy benefit as discussed earlier.

(3) **Gradual Sybil infiltration is *hindered*.** By constraining the graph formation to a much shorter time window of typically less than a minute, an adversary no longer has the time to gradually impersonate Sybils with their malicious nodes. Instead, malicious nodes that impersonate Sybils will pay the price of not being able to validate their own presence as effectively, drawing suspicion upon themselves.

(4) **Listening/broadcasting asymmetry is *accounted for*.** We employ a *directed* graph to represent the connections. Broadcasting a message over WiFi produces a publicly-visible physical footprint in the vicinity, while receiving a message may be done passively. Consequently, **broadcasting serves as much stronger evidence of physical presence than listening**, which will be discussed further in Section 5.1.

(5) **Connection failures are *recorded*.** Connections that *could* have been established between nodes (i.e. one listened while the other broadcasted) but *were not* may be useful in placing doubt on the legitimacy of a node's location claim by serving as **counter-evidence**.

5 PROXIMITY GRAPH ANALYSIS

The core idea for identifying Sybils is that **Sybils are not able to generate *sufficiently-believable* sets of edge outcomes on proximity graphs constructed as per Section 4 due to their inherent physical absence**, even with support from malicious nodes under the three adversarial evasive strategies described in Section 3.4 point (3).

To analyze this concept of *believability of edge outcomes* within the inherently uncertain nature of wireless communication, we introduce a probabilistic model for the communication medium. As the preliminary model used in this paper, we treat the **probability of a successful edge as a function of pairwise distance between the nodes**, *where a listening node has a lower probability of seeing a broadcasting node the further away apart they are.* An example of such a function for WiFi Direct is shown in Fig. 4.

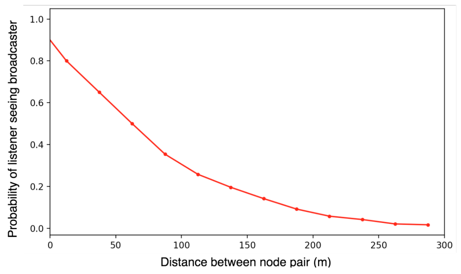

Figure 4: The conditional probability of a listening node *successfully* seeing a broadcasting node's key within $\alpha = 2$ seconds (a single round) using WiFi Direct given the distance between the two nodes (measured experimentally[20]).

The conditional probability of an edge's success is thus assumed to be a function of distance, and is given by $prob : \mathbb{R}^+ \longrightarrow [0, 1]$. To make analysis concrete, we use the experimentally-measured function for WiFi Direct exactly as depicted in Fig. 4. Labelling such a probability of an edge's success as $p_{i,j}$, we have:

$$p_{i,j} = P\Big(res(e_{i,j}) = S \mid dist(e_{i,j})\Big) = prob(dist(e_{i,j}))$$

We may now effectively transform our edge successes and failures into a set of **likelihood** values, one per each directed edge. To do so, we define the function $edgeL : E \longrightarrow [0, 1]$:

$$edgeL(e_{i,j}) \begin{cases} p_{i,j} & \text{if } res(e_{i,j}) = S \\ 1 - p_{i,j} & \text{if } res(e_{i,j}) = F \end{cases}$$

5.1 Determining a Node's Trustworthiness

Directed edges between the honest nodes should succeed/fail randomly according to the *prob* function, and so for every node, **the collection of edge outcomes may be viewed as a realization of a random vector**, and the likelihood of occurrence of *that particular* collection may be assessed.

Without support from malicious nodes, all Sybil–honest connections will fail (as per Section 3.3), and since many of these edges are short, their likelihoods of failure will be low, impacting both the Sybils and the honest nodes alike. The preponderance of honest nodes and the comparative scarcity of Sybils, however, would ultimately result in the Sybils being affected to a far greater degree. This suggests selecting low edge-likelihood nodes to be a reasonable approach to identifying Sybils.

To factor in adversarial evasive strategies into our analysis, we note that for each node in the formed graph, there are *outgoing* edges and *incoming* edges. We argue that the incoming edges constitute a far more resilient dataset than the outgoing edges due to the asymmetric cost for an adversary to forge an incoming (honest→Sybil) edge via IMPERSONATION versus forging an outgoing (Sybil→honest) edge via CIRCULATION. Specifically:

- A single malicious node may *listen* on behalf of *many* Sybils and inform *all* of them about the keys that they should have seen, without cost. Thus it is easy for an adversary to forge successful outgoing edges from Sybils.
- A single malicious node may *broadcast* on behalf of only a *single* Sybil node due to possessing only a single radio transmitter. Thus, the ability of an adversary to contrive successful incoming edges is limited.

Consequently, for a given node v_i, we define its **trustworthiness** level to be the **total observed likelihood $\mathcal{L}^*_{v_i}$** of all of its **incoming edges across all rounds**, where:

$$\mathcal{L}^*_{v_i} = \prod_{j \neq i}^{N} edgeL(e_{j,i})$$

For both theoretical and computational reasons, it proves convenient to work with the natural logarithm of this quantity – the **observed log-likelihood value $\ell^*_{v_i}$** – defined as:

$$\boldsymbol{\ell}^*_{v_i} = ln(\mathcal{L}^*_{v_i}) = \sum_{j \neq i}^{N} ln(edgeL(e_{j,i}))$$

Following up on the earlier example given in Section 4.5 with nodes A, B, C, D where C is a Sybil and the $D \rightarrow A$ edge is the only edge to have failed randomly, the transformed *probabilistic* proximity graph that would be created is depicted by Fig. 5 below.

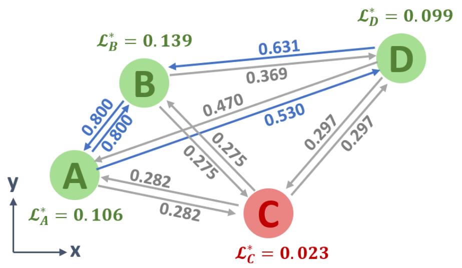
$\mathcal{L}^*_D = 0.099$
$\mathcal{L}^*_B = 0.139$
0.631
0.369
D
B
0.470
0.530
0.800
0.800
0.275
0.275
0.297
0.297
A
0.282
0.282
C
y
x
$\mathcal{L}^*_A = 0.106$
$\mathcal{L}^*_C = 0.023$

Figure 5: *Probabilistic* proximity graph derived by applying the WiFi Direct *prob* function to the scenario from Section 4.5, assuming shown locations to be at scale and with $dist(e_{A,B}, rnd(e_{A,B})) = 12.5m$. Incoming-edge-$\mathcal{L}^*_v$ values are shown for every node; the Sybil's value is noticeably low, and the difference only becomes more extreme with higher quantities of nodes.

5.2 Evaluating a Node's Log-Likelihood

As Sybils are likely to exhibit lower likelihood values (and thereby lower log-likelihood values) for their incoming edge sets, we deem suspicious nodes with low $\boldsymbol{\ell}^*$ values. However, a node's observed $\boldsymbol{\ell}^*$ value is a *realization* of a random vector whose distribution is dependent on the exact scenario of the total number of nodes and their round-wise locations, and consequently a "low" threshold does not have an absolute measure and is graph-dependent.

To find this distribution, we first define binary random variables $X_{j,i}$ each of which has two possible outcomes: the log-likelihood value given a success of edge $e_{j,i}$, and given its failure. While these variables are mutually dependent through the node locations, they become mutually independent once those locations are fixed when the graph is formed. The $X_{j,i}$ are defined as follows:

$$X_{j,i} = \begin{cases} ln(p_{j,i}) & \text{with probability } p_{j,i} \\ ln(1 - p_{j,i}) & \text{with probability } 1 - p_{j,i} \end{cases}$$

The total log-likelihood of all incoming edges into a node is given by the random variable $\boldsymbol{\ell}_{v_i}$, defined as the sum of the individual $X_{j,i}$ variables:

$$\boldsymbol{\ell}_{v_i} = \sum_{j \neq i}^{N} X_{j,i}$$

As such, the observed $\boldsymbol{\ell}^*_{v_i}$ value of node v_i is one particular realization of the $\boldsymbol{\ell}_{v_i}$ random variable. We deem v_i a *Sybil candidate* if the *p-value* ρ_{v_i} of its observed $\boldsymbol{\ell}^*_{v_i}$ is so small that it falls under the 5σ boundary (chosen empirically), namely when:

$$\rho_{v_i} = P(\boldsymbol{\ell}_{v_i} \leq \boldsymbol{\ell}^*_{v_i}) \leq (3.5 * 10^6)^{-1}.$$

The probability density of $\boldsymbol{\ell}_{v_i}$ may be computed directly by evaluating every possible combination of edge outcomes, but this is exponential in nature (of order 2^{N^2}). It is possible to optimize this process losslessly by computing the outcomes in a strict increasing order, terminating early when their combined likelihoods exceed the threshold and checking whether the observed value $\boldsymbol{\ell}^*_{v_i}$ is within the range of this computed left-tail. It is also possible to approximate this computation by quantizing the values of the $X_{j,i}$ variables, allowing them to be bundled together, thereby greatly reducing the number of distinct computations necessary. Neither of these approaches fundamentally alters the exponential runtime however, and thus both remain impractical.

5.3 Lyapunov's CLT Approximation

To obtain the probability density of a particular node's log-likelihood random variable $\boldsymbol{\ell}_{v_i}$ within practical time constraints, we accurately approximate it via the Central Limit Theorem (CLT) **in linear time**. As the random variable $\boldsymbol{\ell}_{v_i}$ is a sum of independent but *differently-distributed* random variables $X_{j,i}$, each with a bounded expectation and bounded variance, we use Lyapunov's version of the CLT to deduce that $\boldsymbol{\ell}_{v_i}$ is approximately normal. For Lyapunov's CLT theorem to apply, it suffices to have the following condition hold for some single value of $\delta > 0$:

$$\frac{\sum_{j \neq i}^{N} \mathbb{E}\left[\left|X_{j,i} - \mathbb{E}[X_{j,i}]\right|^{2+\delta}\right]}{s_N^{2+\delta}} \xrightarrow{N \to \infty} 0$$

We prove that this condition holds for $X_{j,i}$ variables in proximity graphs by setting $\delta = 2$, finding an upper bound on the numerator and a lower bound on the denominator, and showing that this total upper bound on the fraction vanishes at the limit. Note that while the numerator and denominator are not independent, examining these two bounds separately can only yield a tighter bound.

Firstly, the value of each individual summand in the numerator—as function over $p_{j,i}$ values—is shown in the left graph in Fig. 6. Each summand is upper-bound by 5, giving the following upper bound on the sum of N of these terms:

$$\mathbb{E}\left[\left|X_{j,i} - \mathbb{E}[X_{j,i}]\right|^4\right] < 5 \text{ for all } p_{j,i} \in (0, 1)$$

$$\Rightarrow \sum_{j \neq i}^{N} \mathbb{E}\left[\left|X_{j,i} - \mathbb{E}[X_{j,i}]\right|^4\right] < 5N$$

For the variance term in the denominator, we present the following argument. The variance of each individual $X_{j,i}$ term—as a function of $p_{j,i}$—is shown in the right graph in Fig. 6, and is small near $p_{j,i} \in \{0, 0.5, 1\}$.

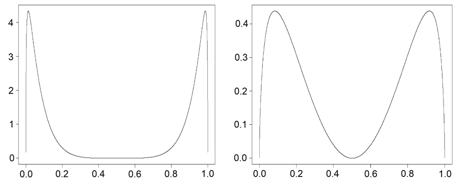

Figure 6: Values of $\mathbb{E}\left[\left|X_{j,i} - \mathbb{E}[X_{j,i}]\right|^4\right]$ (left) and $Var(X_{j,i})$ (right) for possible values of $p_{j,i}$.

By our probability model, $p_{j,i}$ are never 0 or 1. Additionally, as the node locations are confined to a 2-dimensional plane, it is *impossible* to have more than 3 nodes be mutually equidistant, implying that $p_{j,i} = 0.5$ can only hold for *all* nodes if there are 3 or fewer of them in total.

With these spacial constraints, we claim that there is a proportion γ of the N edges that each have individual variances greater than some constant non-zero value η. This gives the following lower bound on the total variance s_N^2:

$$s_N^2 = \sum_{j \neq i}^{N} Var(X_{j,i}) > \gamma N \eta$$

With an upper bound on the numerator and a lower bound on the denominator, we show that the Lyapunov condition is indeed satisfied as the fraction vanishes with increasing N:

$$\frac{\sum_{j \neq i}^{N} \mathbb{E}\left[\left|X_{j,i} - \mathbb{E}[X_{j,i}]\right|^4\right]}{(s_N^2)^2} < \frac{5N}{\gamma^2 N^2 \eta^2} = o\left(\frac{1}{N}\right)$$

With the condition satisfied, we have that the sum of demeaned and normalized $X_{j,i}$ variables is approximately normal:

$$\sum_{j \neq i}^{N} \frac{X_{j,i} - \mathbb{E}[X_{j,i}]}{s_n} \xrightarrow{\mathcal{D}} \mathcal{N}(0,1)$$

We additionally verified this normal approximation empirically to be extremely accurate even with as few as 22 nodes by randomly generating moving nodes, simulating communication, obtaining corresponding $p_{j,i}$ values, and then directly computing the probability densities for each node. Sample probability densities for the log-likelihood of a typical node in models with 14, 18, and 22 total nodes are shown in Fig. 7.

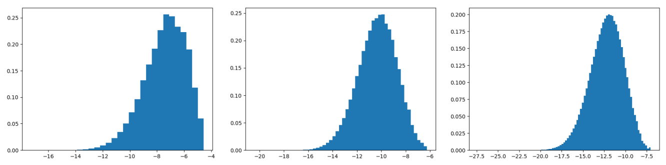

Figure 7: Distribution of summed log-likelihood of a typical node in a model with 14, 18, and 22 nodes respectively (computed directly). Typical scenarios have over 100 nodes, which allow for an even better normal approximation.

5.4 Sybil Identification Procedure

We outline the procedure to obtain the *p-value* for a node's observed log-likelihood value, utilizing the density approximation via Lyapunov's CLT, in the algorithm below. In this algorithm, the observed value, the mean, and the variance of every edge (treated as a random variable) is summed. Then, the left-tail p-value sum of the observed variables is found from a Gaussian parametrized by the mean-sum and variance-sum.

Algorithm 2 Computing the *p-value* of a Node's log-likelihood ℓ^*

```
procedure NODE_LLH_PVAL(i, known_sybils)
    llh ← 0
    mean ← 0
    var ← 0
    for edge in edges_of_node(i) do
        j = source_node_id(edge)
        if $j \notin known\_sybils$ then
            $\mathbb{E}[X] \leftarrow (p_{j,i})ln(p_{j,i}) + (1 - p_{j,i})ln(1 - p_{j,i})$
            $\mathbb{E}[X^2] \leftarrow (p_{j,i})ln(p_{j,i})^2 + (1 - p_{j,i})ln(1 - p_{j,i})^2$
            $llh$ += $ln(edgeL(e_{j,i}))$
            $mean$ += $\mathbb{E}[X]$
            $var$ += $\mathbb{E}[X^2] - \mathbb{E}[X]^2$
    $Z \leftarrow \frac{llh - mean}{\sqrt{var}}$
    pval ← INV_NORM(Z)
    return pval
```

Perhaps the simplest algorithm for identifying Sybils is to simply assume that all Sybil candidates – those nodes with a log-likelihood's *p-value* lower than the 5σ left tail are, in fact, all Sybils. We denote this method as the **Simple Sybil Detection** procedure, and outline the algorithm below.

Algorithm 3 Simple Sybil Detection

```
procedure SIMPLE_SYBIL_DETECTION
    sybils ← []
    for j = 1 ... N do
        pval ← NODE_LLH_PVAL(i, [])
        if $pval \leq (3.5 * 10^6)^{-1}$ then
            sybils.APPEND(i)
    return sybils
```

While our simulation results show that this procedure is actually very effective in practice, it nonetheless disregards the reinforcing effect Sybils have on each others' *p-values* and similarly their diminishing effect on honest nodes' *p-values*. To account for this compounding effect, we propose the **Iterative Sybil Detection** procedure.

Algorithm 4 Iterative Sybil Detection

```
procedure ITERATIVE_SYBIL_DETECTION
    sybils ← []
    while true do
        pvals ← []
        for i = 1 ... N do
            pvals[i] ← NODE_LLH_PVAL(i, sybils)
        min ← ARGMIN(pvals)
        if $pval[min] \leq (3.5 * 10^6)^{-1}$ then
            sybils.APPEND(min)
        else
            return sybils
```

In this procedure, we perform a sequence of *p-value* computations for all nodes, declaring the single most suspicious node as a Sybil in every iteration and removing it from all further *p-value* computations for all subsequent iterations. In this manner, we crumble the entire Sybil network one Sybil at a time, as every Sybil typically becomes increasingly more suspicious the fewer of them remain. We demonstrate the superior performance of this technique in Section 6.

6 SIMULATION OVERVIEW AND RESULTS

In order to examine the efficacy of our model, we built a to-scale simulator that mimics the entire validation process (in python), from generating the nodes, to generating a communications plan, simulating its execution, employing the adversarial obfuscation techniques, and analyzing the created proximity graph. The simulator is publicly available on github[2] and uses the SyPy library[12] for comparative graph-detection methods. Each simulation run has the following flow:

(1) Terrain layout is generated.
(2) Nodes can either be imported or generated using a simple spawner which spawns clusters of specified dimensions, locations, and node types ('hon', 'syb', 'mal'). Spawned nodes have speeds sampled from $\mathcal{N}(1.3ms^{-1}, 0.2ms^{-1})$[15].
(3) A set of nodewise state sequences is generated, detailing whether a given node in a given round: is idle, listens, or broadcasts a unique key.
   * **Impersonation**: In every round, a subset of listening malicious nodes instead broadcasts on behalf of Sybils.
(4) In every round, for every listener–broadcaster pair, a random binary outcome is generated as described in Section 5. If the outcome succeeds **or** if this is a Sybil–Sybil/Sybil–malicious pair, broadcaster's key is added to listener's *seen list*.
   * **Circulation**: In every round, malicious nodes that *actually* listened disseminate their *seen keys* to all other malicious and Sybil nodes who were *supposed* to be listening.
   * **Dissociation**: Malicious and Sybil nodes randomly drop each other's keys from their *seen lists* with probability D.
(5) In every round, edges are created from every listener to every broadcaster. Each such edge is labelled *successful* if the listener's *seen list* contains the corresponding broadcaster's key, and *failed* otherwise. This forms the proximity graph.
(6) Sybil detection is performed on the proximity graph (the set of nodes and the set of successful and failed edges).

6.1 Target Application

For our target application, we chose the explosive-detection sensor network from [14]. This led to the natural choice of WiFi Direct as the communication medium, along with a radius $\mathcal{R} = 32.5m$, allowing for two antipodal nodes—moving in opposite directions with typical walking speeds of $1.3ms^{-1}$—to remain within $200m$ of each other over 45 seconds. The round duration was set to $\alpha = 2$ seconds, because this is sufficient to broadcast and scan the WiFi Direct band[21], and also for consistency with the available experimental results[20] of WiFi connection probability (Fig. 4). The Sybilness threshold value remained 5σ. In other words, Sybil candidates were nodes whose log-likelihood of incoming edges is roughly within the lowest $(3.5 * 10^6)^{-1}$ of least-probable log-likelihood values.

For the scenarios, we model a street with wide sidewalks with all nodes constrained to horizontal movement along them, as can be seen in Fig. 8. The dimensions are roughly modeled off Market Street in San Francisco.

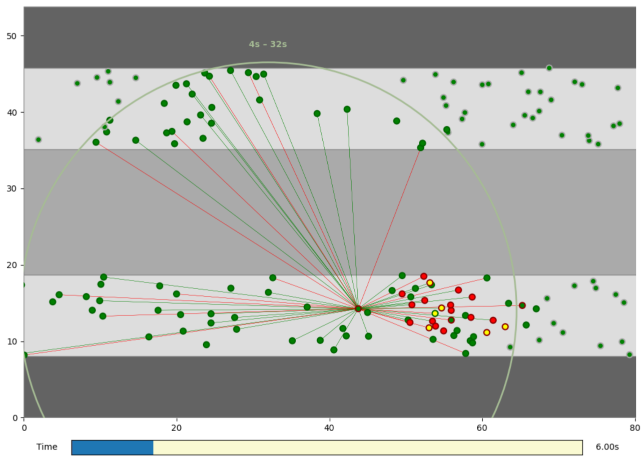

Figure 8: Simulator GUI, showing a run 6 seconds into the communication procedure (large light-green circle). Nodes are color coded according to their true type (*fill* color: green for honest, yellow for malicious, red for Sybil) and predicted type (*edge* color: red for Sybil, green for non-Sybil, grey for non-participant). The lines depict successful (green) and failed (red) edges that were attempted at time=6sec for the misclassified honest node. All distances are to scale (metric).

6.2 Detection Algorithms and Methodology

We perform Sybil detection with the following four algorithms, the former two being our work (described in Section 5) and the latter two being GSD solutions:

(1) Simple Sybil Detection
(2) Iterative Sybil Detection
(3) SybilPredict
(4) SybilRank

As our target application was the explosive-detection system, we deemed the most important quality metric of any detection algorithm to be the recall rate, corresponding to the fraction of total Sybils detected, and a secondary metric to be precision (true positive rate) indicating the amount of honest nodes misclassified as Sybils.

For each data point in all subsequent graphs, we ran 20 scenario simulations, giving 20 different sets of node positions and momenta. For *each* such scenario simulation, we ran 20 simulations of proximity graph construction as described in Section 4 (using **all** of the nodes), with edge successes randomly generated according to the aforementioned WiFi Direct function[20]. The subsequent plots

[2]https://github.com/nverchok/HuntingSybils

detail the means and standard deviation of these 400 recall and precision values *per* data point in every plot.

Additionally, malicious nodes were completely ignored from calculating the precision and recall values, as it is unclear how to classify them. On the one hand, they are adversarial and so detecting them is good. On the other, they are physically-real and therefore should *not* be identified as Sybils.

6.3 Detection: Sybils Only

The first set of simulations we examined consisted of Sybils exclusively, without any malicious nodes at all, resulting in Sybils being utterly unable to form any connections from any honest nodes at all. From the perspective of the Simple and Iterative detection algorithms, all Sybil log-likelihood values were therefore extremely low as the Sybils simply had too many failed short edges. From the perspective of GSD algorithms, the lack of any successful edges between Sybils and honest nodes resulted in obvious Sybil communities. Consequently, all 4 detection algorithms consistently detected all Sybils with virtually no false negatives.

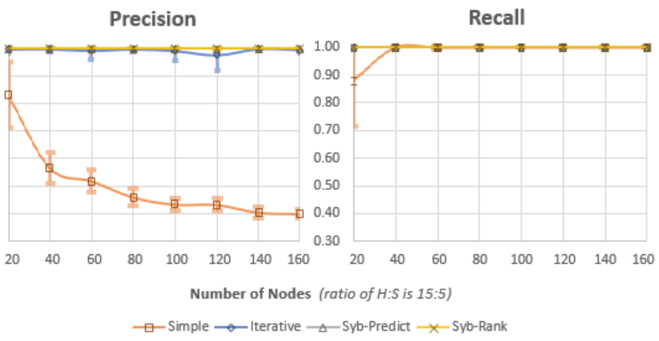

Figure 9: Detection results with only honest and (disconnected) Sybil nodes. Error bars indicate standard deviation.

As expected, however, the Simple Sybil detection algorithm produced a significant number of false positives (honest nodes detected as Sybils; *precision*) as seen in Fig. 9 above, since the honest nodes had their log-likelihoods severely reduced by the many failed short edges from the Sybils. In contrast, the Iterative scheme was able to gracefully handle such situations by plucking Sybils one at a time as shown in Fig. 10, throwing them out of subsequent p-value computations, and thereby resulting in reasonable p-values for most honest nodes. The GSD algorithms naturally had zero false positives for this trivial disconnected graph.

6.4 Detection: No Evasive Strategies

For the next set of simulations, we introduced malicious nodes but did not yet enable any *evasive strategies*, meaning that Impersonation, Circulation, and Dissociation were **not** employed by the Adversary. Thus Sybil–Sybil and Sybil–malicious edges were *always* established irrespective of distance, while connections between physically-real nodes (honest and malicious) were generated according to the probability–distance function as described in Section 5 as per normal. This resulted in very obvious Sybil

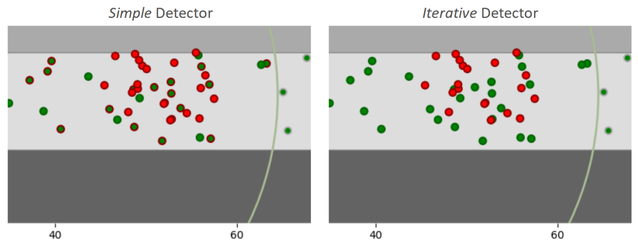

Figure 10: The Simple detector has a significant false positive rate as the honest nodes are made suspicious by virtue of the nearby Sybils. As the Iterative detector sequentially flags and discards the most suspicious nodes, it is able to pluck apart the Sybil cluster node by node, which in turn greatly lowers the false positive rate (zero in this simulation).

clustering from the GSD perspective, and in low log-likelihood values from the Simple and Iterative perspective, since the perfect inter-connectivity was still not enough to overcome all of the failed Sybil–honest edges. The results are given by Fig. 11 below.

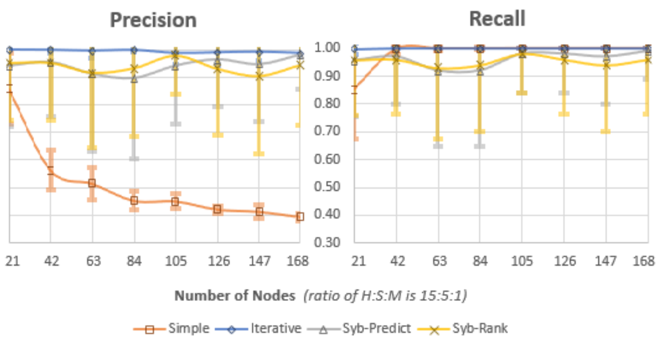

Figure 11: Detection results with no evasive strategy (obvious clustering). Error bars indicate standard deviation.

The high recall averages in 0.9+ range show that, without an obfuscation strategy, the Sybils were still extremely easy to detect despite their slight connectedness to the honest nodes (through the malicious nodes) using any of the 4 algorithms. The near-zero recall *standard deviations* (error bars) for Iterative Detection and Simple Detection demonstrate their incredible consistency, while the recall standard deviations for the two GSD algorithms are much higher in the 0.2 to 0.3 range. These higher standard deviations–stemming exclusively from **catastrophic misclassification outcomes**–highlight the fragility of GSD methods and the comparative robustness of our approach.

The precision figures also demonstrate several interesting characteristics. Again, the average precision of Simple Detection is very low, dipping below 0.4 at higher quantities of nodes. This is once again caused by the adverse effect of disconnected Sybils on nearby honest nodes. The Iterative Detection algorithm can once again be seen to successfully rectify this issue however, achieving precision values *consistently* near 1.0 with virtually no standard deviation. By contrast, while the average precision values for the GSD algorithms are also high in the 0.9+ range, they all have significant standard deviations in the 0.2 range, again as a result of occasional catastrophic misclassification.

6.5 Detection: Smart Sybil Obfuscation

For the next set of simulations, we introduced the 3 obfuscation techniques discussed prior. For Impersonation we used one single dedicated malicious listener (per round), with all other malicious listeners broadcasting randomly-chosen Sybil keys instead. For Circulation, the single malicious listener in every round spread all of its seen keys to all malicious and Sybil nodes that were supposed to be listening in that round. For Dissociation, Sybil–Sybil and Sybil–malicious connections were formed according to the WiFi Direct distance–probability function (the assumption being that the Adversary has full knowledge of our evaluation function), but the Sybil–Sybil success probability was downscaled using (multiplied by) the parameter $D \in \{0.5, 0.4, 0.2\}$ in order to break up the Sybil community. The results across the three values of the Dissemination parameter D are given by Fig. 12 below.

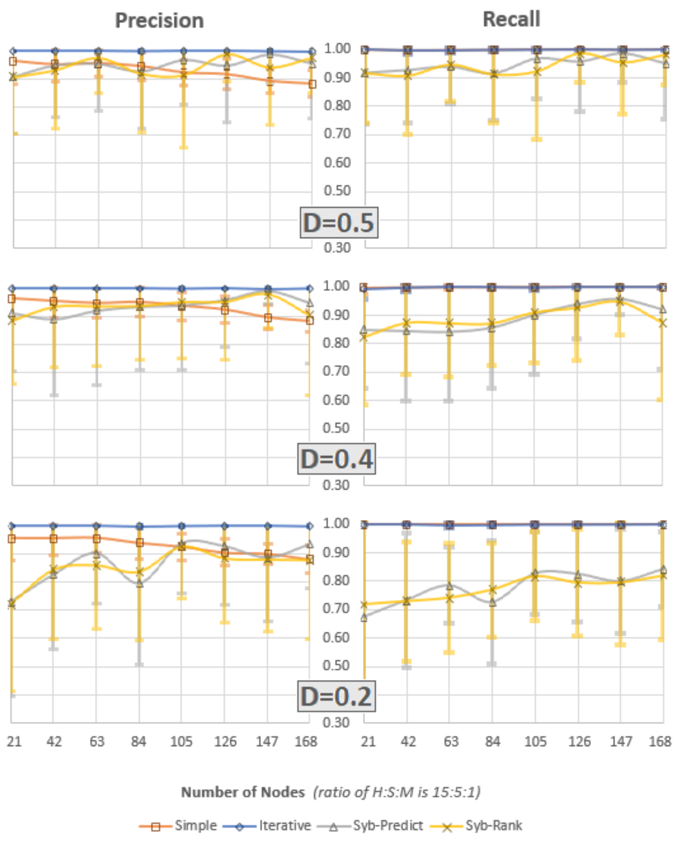

Figure 12: Detection performance results with an evasive strategy consisting of the Impersonation, Circulation, and Dissociation (across three Dissociation parameters $D \in \{0.5, 0.4, 0.2\}$). Error bars indicate standard deviation.

The Iterative approach still worked just as effectively as before, since the amount of roundwise Sybil broadcasts that could be impersonated was limited. Most Sybils were still just as suspicious as in the scenarios without any evasive strategy, and were easily pruned out due to the low p-values of their respective log-likelihoods. The Sybils that had been integrated successfully gradually lost the support of the un-impersonated Sybil majority that was pruned out, and the few rounds wherein they were impersonated yielded insufficient credibility to overcome detection. Interestingly, the precision of Simple detection improved over the previous set of results due to the successful Sybil–honest edges formed via Circulation.

The major point of interest in these results was the steadily declining recall performance of GSD algorithms with decreasing values of the Dissociation parameter D, dropping to the 0.8 range. This is directly due to the 3 obfuscation techniques, which created Sybil–honest edges while reducing the quantity of Sybil–Sybil and Sybil–malicious edges. We showcase the effects of Impersonation and Dissociation in Fig. 13 in a catastrophic misclassification outcome, where the incoming edges into two Sybil nodes misclassified by Sybil Rank are displayed.

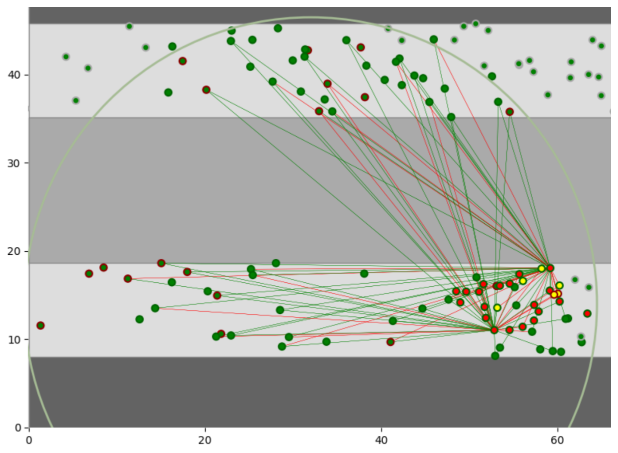

Figure 13: A simulation run that resulted in catastrophic misclassification by both the Sybil Rank and Sybil Predict algorithms, detecting no Sybils and instead misclassifying honest nodes. The failure is due to adversarial Impersonation and Dissociation, which de-cluster the Sybils by failing some Sybil→Sybil edges and forming some honest→Sybil edges, as displayed for two particular Sybils.

The GSD algorithms once again produced much higher standard deviations for both precision and recall, again showing their brittleness when applied to a probabilistic proximity graph, resulting from the fact that they do *not* incorporate the likelihoods of edge successes and failures.

6.6 Detection: Increasing Sybil Proportion

For our final assessment of the robustness of the detection schemes, we kept constant the numbers of honest and malicious nodes at 100 and 5 respectively, gradually increasing the number of Sybils. The increasing Sybil-to-malicious ratio meant that a smaller proportion of Sybils could be IMPERSONATED, reducing the extent to which the Sybils could be intermingled with the honest nodes. However the increasing Sybil-to-honest ratio negatively impacted the Simple and Iterative Detection algorithms due to the adverse effect of Sybils on honest nodes' log-likelihoods. The results are given in Fig. 14.

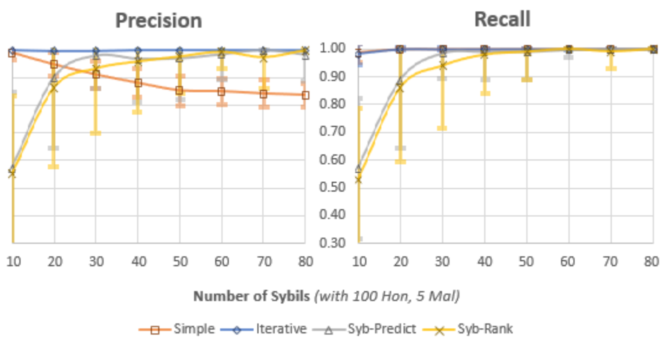

Figure 14: Detection performance results with 100 honest and 5 malicious nodes, and an increasing number of Sybils. The evasive strategy again consists of IMPERSONATION, CIRCULATION, and DISSOCIATION (with $D = 0.5$).

As expected, the GSD algorithms showed improvement as the Sybil proportion increased. Simple Detection still consistently produced some false positives (an amount increasing with the increasing number of Sybils), but Iterative Detection was once again robustly producing perfectly-consistent 1.0 recall values.

7 CONCLUSION

In this paper, we have outlined the current technical challenges to Sybil countermeasures within Cyber-Physical Systems. We have discussed the impracticality of the current proximity graph construction due to its slow nature that hogs the device radio, its reliance on gradually-accumulated historical data, and its susceptibility to gradual Sybil infiltration, posing a major security threat.

To simultaneously address all of these issues, we proposed a novel logarithmically-fast, one-shot proximity-graph construction scheme which drastically reduces the time required for devices to participate, requires no prior historical data, and yet also imposes a credibility cost for malicious nodes that try to impersonate Sybils, hindering Sybil infiltration.

We have then explained how this new *probabilistic proximity graph* with *uncertain* edges (stemming from imperfect connectivity) produces very brittle results when analyzed using traditional GSD-based algorithms. Furthermore, we have demonstrated how Sybils can collude with a small number of *malicious* nodes, employing the IMPERSONATION, CIRCULATION, and DISSOCIATION techniques to bypass detection by GSD algorithms. We have argued that this new proximity graph inherently encodes information not exploited by GSD algorithms, developed a novel mathematical framework for node-trustworthiness assessment, and proposed our own algorithms for Sybil detection.

We have then applied this entire scheme to a particular participatory explosive-detection system by building a simulator and have demonstrated the efficacy of our approach. In particular, throughout all of our simulations, the proposed Iterative Detection scheme consistently achieves near 1.0 recall values with near 0.0 standard deviation, while the two GSD algorithms examined succumb to the three obfuscation techniques and consistently produce non-negligible amounts of catastrophic misclassifications.

We would like to emphasize the role of our approach as a foundational framework which not only generalizes to a vast class of P2P-capable Cyber-Physical Systems, but is also greatly extensible. In our preliminary model, we determined the success probability of connections as a function of pairwise distances, but our framework can easily incorporate more sophisticated and realistically-accurate models of connectivity.

ACKNOWLEDGMENTS

The authors gratefully acknowledge the support of the NSF grant CPS-1739684.

REFERENCES

[1] N. D. Lane, E. Miluzzo, H. Lu, D. Peebles, T. Choudhury and A. T. Campbell, A survey of mobile phone sensing, in IEEE Communications Magazine, 2010.
[2] T. Monahan, J. T. Mokos, Crowdsourcing urban surveillance: The development of homeland security markets for environmental sensor networks, Geoforum, 2013.
[3] J. A. Burke, D. Estrin, M. Hansen, A. Parker, N. Ramanathan, S. Reddy, and M. B. Srivastava, Participatory Sensing, UCLA: Center for Embedded Network Sensing, 2006.
[4] B. Guo, Z. Yu, X. Zhou and D. Zhang, From Participatory Sensing to Mobile Crowd Sensing, IEEE International Conference on Pervasive Computing and Communication Workshops (PERCOM WORKSHOPS), Budapest, 2014.
[5] D. Christin, A. Reinhardt, S. S. Kanhere, M. Hollick, A survey on privacy in mobile participatory sensing applications, Journal of Systems and Software, 2011.
[6] K. Shilton. Four billion little brothers?: privacy, mobile phones, and ubiquitous data collection. Commun. ACM 52, 2009.
[7] G. Wang, B. Wang, T. Wang, A. Nika, H. Zheng and B. Y. Zhao, Ghost Riders: Sybil Attacks on Crowdsourced Mobile Mapping Services, in IEEE/ACM Transactions on Networking, 2018.
[8] J. R. Douceur, The sybil attack, in IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems, UK: Springer-Verlag, 2002.
[9] S. Kar and J. M. F. Moura, Distributed Consensus Algorithms in Sensor Networks With Imperfect Communication: Link Failures and Channel Noise, in IEEE Transactions on Signal Processing, 2009.
[10] H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman, SybilGuard: defending against sybil attacks via social networks, In Proceedings of SIGCOMM '06, ACM, New York, 2006.
[11] Q. Cao, M. Sirivianos, X. Yang, and T. Pregueiro, Aiding the detection of fake accounts in large scale social online services, In Proceedings of NSDI'12, USENIX Association, Berkeley, 2012.
[12] Y. Boshmaf, K. Beznosov and M. Ripeanu, Graph-based Sybil Detection in social and information systems, IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2013), Niagara Falls, 2013.
[13] U.S. Department of Homeland Security, DHS/SI&T/PIA-021 Cell All, 2011.
[14] D. Badawi, S. Ozev, J. B. Christen, C. Yang, A. Orailoglu, A. E. Çetin, Detecting Gas Vapor Leaks Through Uncalibrated Sensor Based CPS, ICASSP, 2019.
[15] S. Chandra and A. Bharti, Speed Distribution Curves for Pedestrians During Walking and Crossing, Procedia - Social and Behavioral Sciences, 2013.
[16] K. Mekliche and S. Moussaoui, L-P2DSA: Location-based privacy-preserving detection of Sybil attacks, 11th International Symposium on Programming and Systems (ISPS), Algiers, 2013.
[17] M. Demirbas, Y. Song, An RSSI-based Scheme for Sybil Attack Detection in Wireless Sensor Networks, In Proceedings of International Symposium on a World of Wireless, Mobile and Multimedia Networks, 2006.
[18] K. Rabieh, M. M. E. A. Mahmoud, T. N. Guo and M. Younis, Cross-layer scheme for detecting large-scale colluding Sybil attack in VANETs, IEEE International Conference on Communications (ICC), London, 2015.
[19] Gang Wang, Bolun Wang, Tianyi Wang, Ana Nika, Haitao Zheng, and Ben Y. Zhao, Poster: Defending against Sybil Devices in Crowdsourced Mapping Services, In Proceedings of MobiSys '16 Companion, ACM, New York, 2016.
[20] Calvin Newport, David Kotz, Yougu Yuan, Robert S. Gray, Jason Liu, and Chip Elliott, Experimental Evaluation of Wireless Simulation Assumptions, Simulation, 2007.
[21] D. Camps-Mur, A. Garcia-Saavedra and P. Serrano, Device-to-device communications with Wi-Fi Direct: overview and experimentation, in IEEE Wireless Communications, 2013.